PORTFOLIO 2026

김민욱

BACKEND ENGINEER

email. kmwpcpk@gmail.com

tel. 010-3389-6730

github. github.com/kr-Uk

김민욱

백엔드 개발자



name. 김민욱

email. kmwpcpk@gmail.com

tel. 010-3389-6730

github. github.com/kr-Uk

about me

최선을 기대하되, 최악에 대비하는 백엔드 개발자입니다.

사용자를 머무르게 하는 것은 화려한 기능이 아니라 무너지지 않는 신뢰성이라고 생각합니다. 그래서 제가 짠 코드도 언젠가 실패할 수 있다고 전제하고, 정상 흐름보다 무너지는 순간을 먼저 상상하며 그 지점부터 설계를 시작합니다.

project summary

1. DotShelf

데스크톱 독서 기록 + 가상 책장 꾸미기 서비스

인프라 총괄 · 백엔드 구조 설계 (6인, BE 2)

2. Itto

위치 기반 소셜 매칭 서비스

Firebase 중심 백엔드를 Spring Boot로 전환 (1인)

3. BoneToBe

AI 골격 분석 기반 의류 추천 서비스

커뮤니티 API · 애플리케이션 메트릭 환경 구축 (6인, BE 3)

education

2025.07 ~ 2026.06 삼성청년 SW·AI 아카데미 14기
(SSAFY 모바일 트랙)

- Spring Boot 기반 REST API 설계
- Android 앱 개발 및 백엔드 연동
- Git, Jira 기반 협업 및 프로젝트

2018.02 ~ 2025.02 한동대학교 컴퓨터공학 전공

certifications

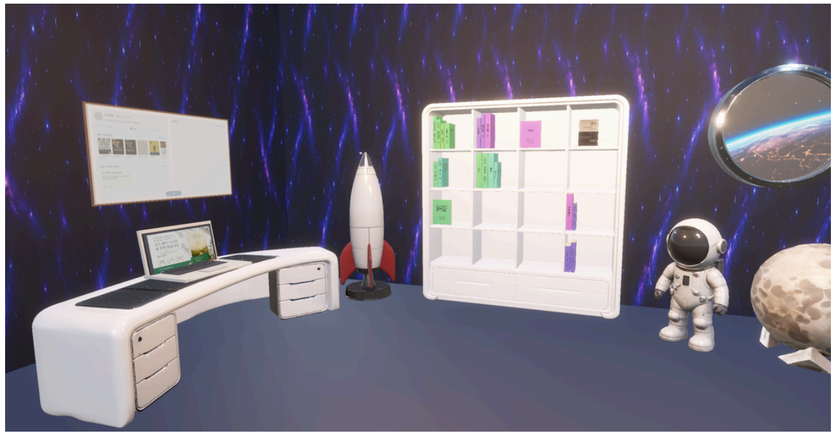
2026.06	SQLD (SQL 개발자)
2026.06	ADsP (데이터 분석 준전문가)
2025.06	정보처리기사
2024.11	OPIc (IM2)

skills

Java · Spring Boot · PostgreSQL · Amazon AWS · Docker · Redis

DotShelf

멀티플랫폼 독서 기록·가상 책장 서비스





개요

GitHub ↗ Demo ↗

읽은 책과 독서 기록을 관리하고, 가상 책장과 독서 공간을 취향에 맞게 꾸며 다른 사용자와 공유하는 독서 기록 서비스입니다.

개발 기간 2026.04 ~ 2026.05 (약 2개월)

개발 인원 팀 프로젝트 6명 (백엔드 2명 · 프론트/Unity 등 4명)

협업 방식 GitLab·Notion MCP 연동: 컨벤션·API 명세·작업 플로우를 AI에 공유해 구현 초안 작성과 1차 코드 리뷰에 활용

담당 **Blue-Green 배포 · 메트릭/로그 모니터링 · S3 업로드 검증**

담당 기술 스택

Spring Boot · JPA · PostgreSQL

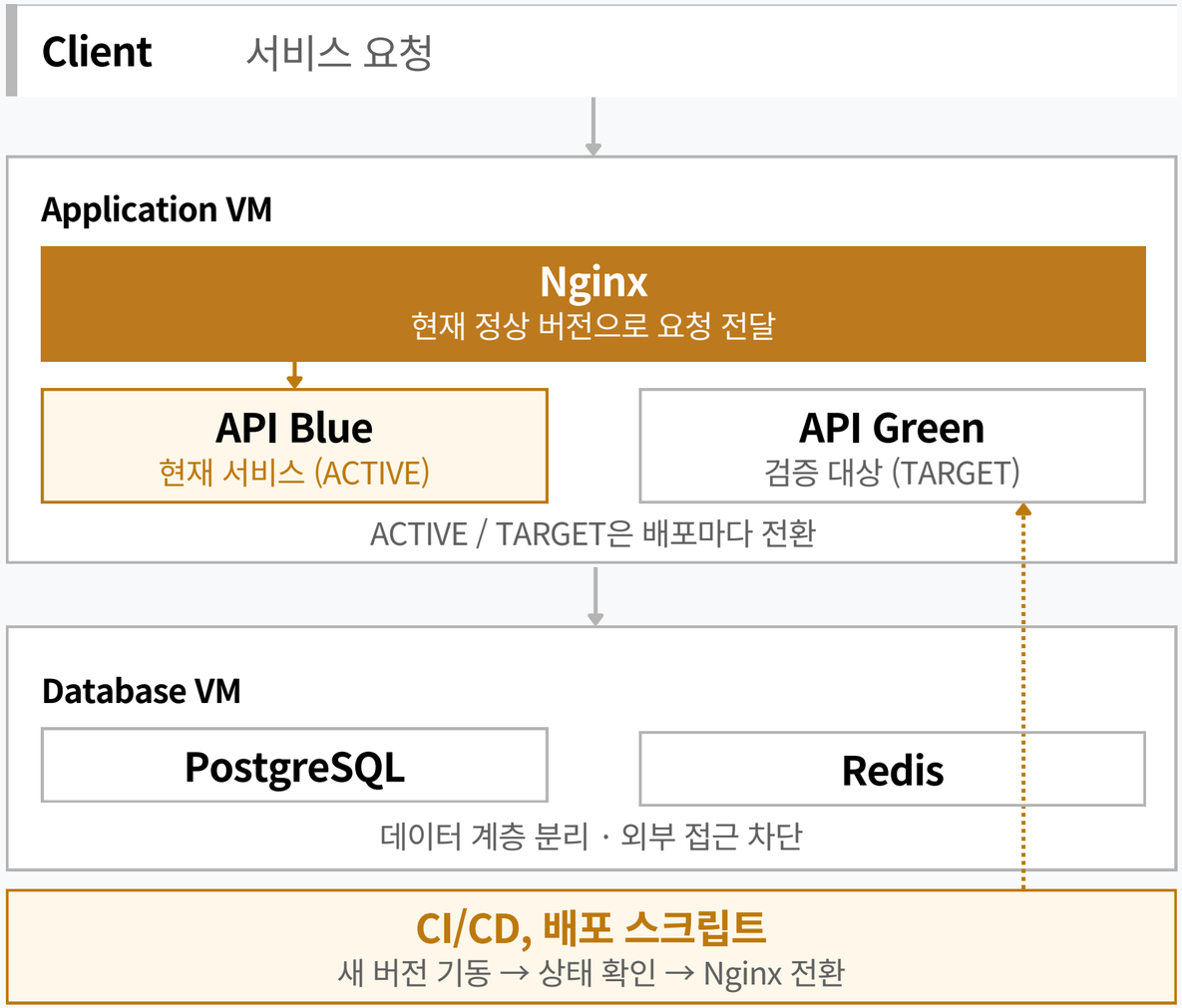
- Spring Boot·JPA 기반 REST API 구현
- PostgreSQL을 사용한 데이터 저장·조회

Docker Compose · Nginx · AWS S3 · GitLab CI/CD · Prometheus · Grafana

- Docker Compose·Nginx로 Blue-Green 컨테이너 운영
- GitLab CI/CD에서 health·smoke 검증 후 Nginx 전환
- Prometheus·Grafana·Loki로 Blue/Green 메트릭·로그 확인
- S3 Presigned PUT + Confirm으로 실제 객체 검증 후 DB 반영

아키텍처

배포 인프라 설계 서비스를 유지한 채 새 버전을 검증하는 구조



담당: Nginx 전환 · 배포/롤백 스크립트 · health 및 smoke 검증

핵심구성

1 **Nginx**
사용자 요청을 현재 정상 서비스로 전달

2 **현재 서비스 / 검증 대상**
기존 버전은 요청을 처리하고,
새 버전은 별도로 기동·검증

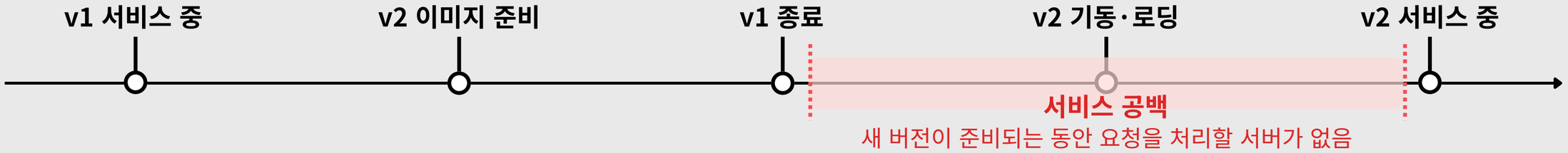
3 **공용 데이터**
Blue와 Green이 함께 사용하는
PostgreSQL · Redis

기능 01 · Blue-Green 배포

서비스를 멈추지 않고 새 버전 배포

새 버전을 먼저 준비하고 검증한 뒤, 요청 대상을 전환했습니다.

일반 배포



Blue-Green 배포



선택한 이유

기존 버전 유지

새 버전 배포 중에도
현재 서비스가 계속 동작합니다.

전환 전 검증

별도 환경에서 Health·Smoke 검증 후
문제가 없을 때만 전환합니다.

전환 후 문제 발생 시 수동 복구

이전 색을 기동하고 Nginx 대상을
수동으로 되돌립니다.

기능 02 · 메트릭/로그 모니터링

배포 후, 새 버전의 상태와 오류를 한 화면에서 확인

Blue와 Green을 구분해 지표로 상태를 보고, 로그로 원인을 확인했습니다.

필요성

메트릭

요청량 · 지연시간 · 5xx · JVM 메모리

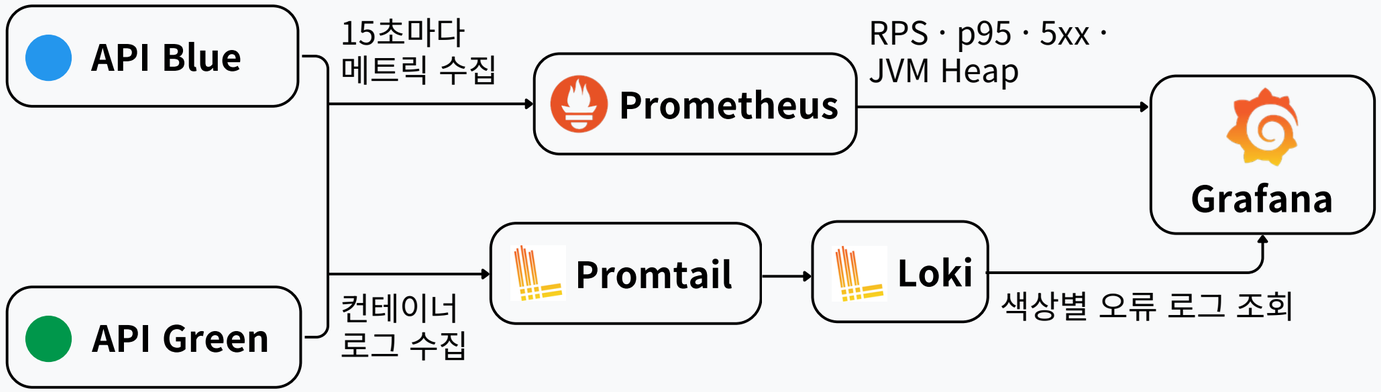
서비스 상태가 어떤지 확인

로그

오류 메시지 · 발생 시점 · 스택트레이스

문제가 왜 생겼는지 확인

모니터링 흐름



숫자로 이상을 찾고, 로그로 원인을 추적

1. 새 버전 기동
2. Blue · Green 지표 비교
3. 이상 시 해당 색상 로그 확인

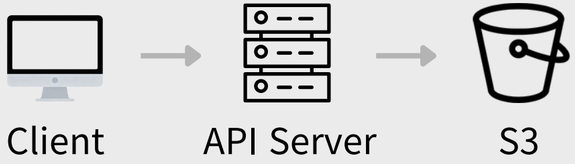
기능 03 · S3 직접 업로드 검증

서버를 거치지 않아도, 확인된 파일만 저장

파일은 S3로 직접 전송하고, 실제 업로드 여부와 메타데이터를 확인한 뒤 DB에 반영했습니다.

직접 업로드의 필요성

서버 경유 업로드



대용량 파일까지 서버가 전달

S3 직접 업로드



파일 전송 부하를 서버에서 분리

단, 직접 업로드만으로는 DB에 기록하기 전 실제 파일을 확인할 수 없음.

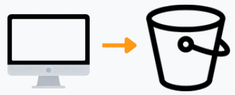
업로드 확정 흐름

1 **업로드 URL 발급**



API가 key·파일형식·크기를 포함한 Presigned URL 발급

2 **S3 직접 업로드**



Client가 S3로 파일 전송

3 **업로드 확인 요청**



Client가 key로 Confirm 요청

4 **S3 HeadObject 검증**



객체 존재 ·Type·Length 확인

5 **DB 반영**



검증 통과한 파일 정보만 저장

검증의 기준

사용자·도메인 key 경로 확인

요청 key가 현재 사용자와 도메인의 허용 경로에 맞는지 확인합니다.

실제 객체가 존재하는지

S3에 해당 key의 객체가 존재하는지 확인합니다.

객체 메타데이터 확인

S3 객체의 Content-Type과 Content-Length가 허용 범위인지 확인합니다.

Itto

스친 인연을, 이토로 잇다

위치 기반 소셜 앱

개요

GitHub · Backend ↗ | Demo ↗

반경 100m 안에서 마주친 사람들의 스침을 기록하고, 반복된 마주침을 관계로 발전시키는 위치 기반 소셜 플랫폼입니다.

개발 기간 2026.02 ~ 2026.06 (5개월)

개발 인원 기획자 1명 + 개발 1인

담당 **Firebase 중심 백엔드를 Spring Boot로 전환**
계층 구조 재정리 · H3 기반 후보군 축소 · 배치 조회 최적화

담당 기술 스택

Spring Boot · JPA · Postgre SQL · Redis · Uber H3

- 매칭 규칙이 늘어나며 복잡해진 도메인 로직을 역할별로 정리
- Redis에 최근 위치를 저장하고, H3 셀로 스침 후보군을 먼저 축소
- 셀 내부 후보 비교의 반복 조회를 일괄 조회와 Map 접근으로 개선

Docker Compose · Nginx · GitHub Actions

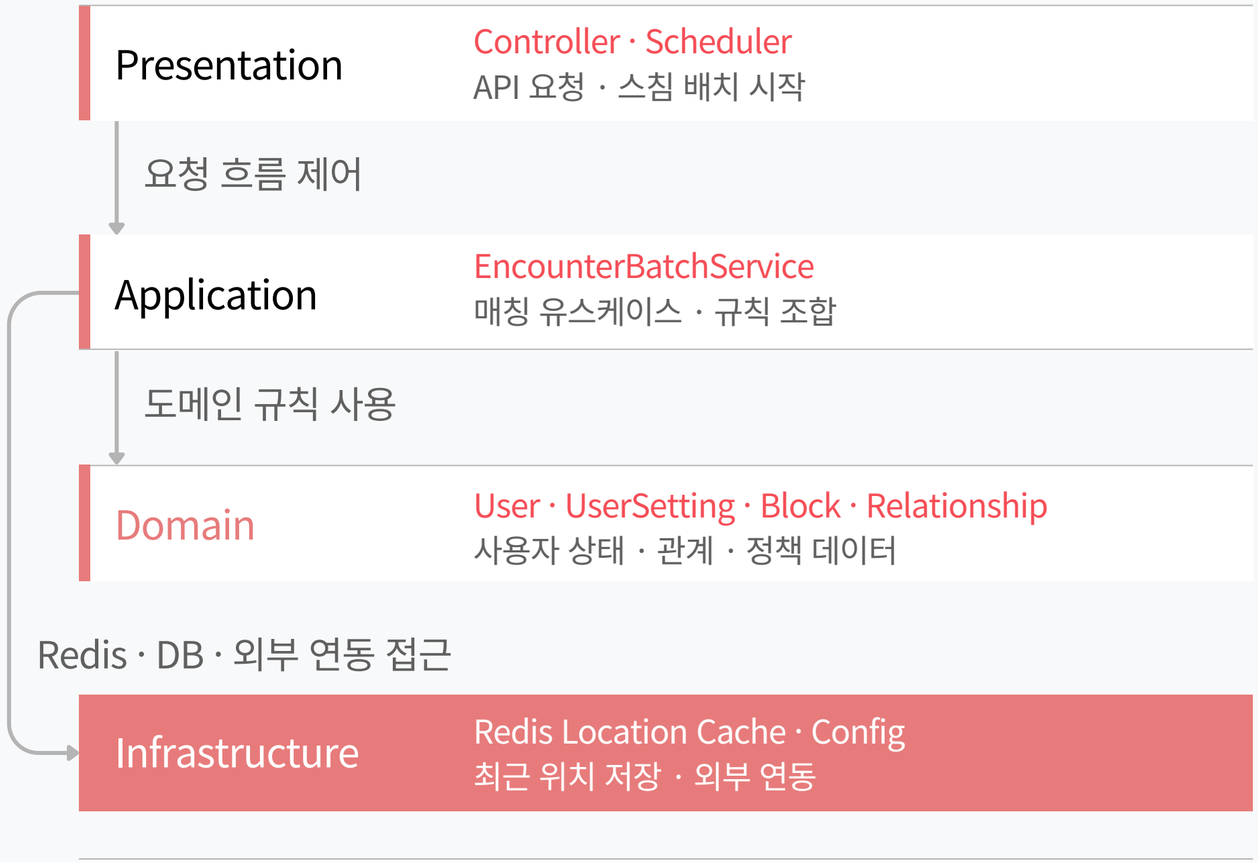
- Docker Compose · Nginx · GitHub Actions 기반 배포 환경 구성



아키텍처

Firebase 중심 백엔드를 Spring Boot로 전환

위치 갱신과 후보 조회가 반복되는 매칭 흐름을 서버에서 직접 통제하도록 구조를 전환했습니다.



Firebase가 맡던 역할

- 인증 · Firestore 데이터 저장
- Firestore Rules 기반 요청별 접근 제어

전환이 필요해진 지점

- 위치 갱신과 후보 조회가 반복되어 Firestore read/write 사용량 증가 우려
- 차단 · 설정 · 기존 관계 · 거리 조건을 함께 판단
- 조건 통과 시 관계와 스침 기록을 한 흐름에서 처리

전환의 기준

위치 갱신과 후보 조회마다 읽기·쓰기 비용이 발생해, 사용량에 따른 비용을 예측하기 어려웠습니다. 또한 차단·관계·거리 같은 복합 조건은 Rules만으로 다루기에는 한계가 있었습니다. 매칭 절차를 서버에서 직접 처리하고 Redis 캐시를 활용해, 비용 구조와 규칙 처리를 모두 통제할 수 있게 했습니다.

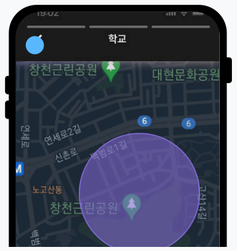
기능 · H3 기반 스침 후보군 축소

전수 비교 전에 가까운 사용자만 후보로 좁히는 방법

위치를 H3 셀 단위로 묶고, 같은 셀 안의 사용자만 실제 거리 계산 대상으로 비교했습니다.

Step 01

위치 수신



서버가 위도 · 경도로
H3 Index 계산

Step 02

Redis 저장



location:user /
location:h3:{cell}

Step 03

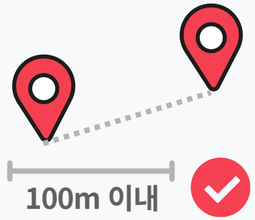
후보군 축소



배치가 활성 H3 셀을 스캔
같은 셀 사용자만 페어 생성

Step 04

정확한 거리 검증



Haversine으로
100m 이내 판정

한계: 현재 배치는 같은 셀만 비교해, 셀 경계 양쪽의 100m 이내 사용자 쌍은 누락될 수 있습니다.

전체 사용자 비교의 한계

- 활성 사용자가 늘수록 전체 사용자 쌍 비교가 빠르게 증가
- 각 사용자 쌍마다 차단 · 기존 관계 · 성별 · 연령대 설정 등 다양한 조건 확인 필요
- 실제 거리 계산과 스침 기록 처리 전, 먼 사용자는 후보 단계에서 먼저 제외할 필요

선택한 방법

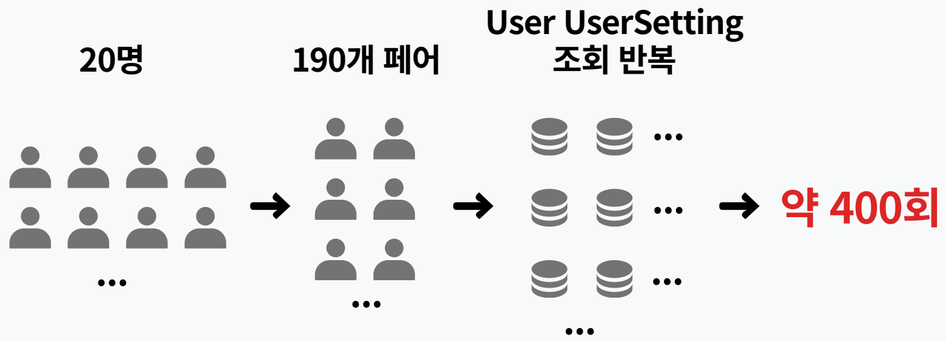
- H3 Resolution 9으로 위치를 셀 단위로 분류
- 셀별 Set에 사용자 ID를 저장해 후보를 먼저 축소
- H3는 후보 탐색용, 최종 판정은 실제 거리 계산으로 분리

트러블슈팅 · 스침 배치 조회 최적화

반복 조회 없이 셀 안의 후보를 비교

사용자와 설정 정보를 셀 단위로 한 번에 조회하고, 이후 비교에서는 Map에서 재사용하도록 개선했습니다.

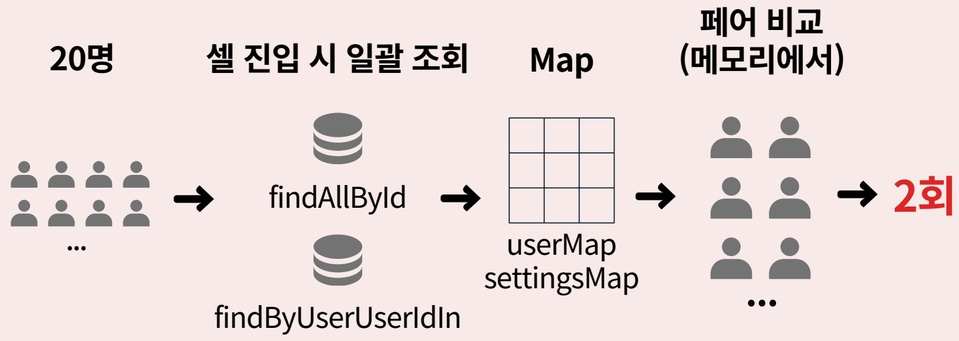
개선 전 · 페어마다 조회



반복문 안에서 사용자 · 설정 조회

반복 조회를
루프 밖으로 이동

개선 후 · 셀 진입 시 일괄 조회



조회는 한 번, 비교는 메모리 Map에서

원인과 해결

- **원인:** 페어 비교 루프 안에서 사용자 · 설정 정보를 재조회
- **해결:** 셀 단위로 일괄 조회 후 userMap · settingsMap 구성
- **검증:** 10 · 20 · 50명 셀에서 User · UserSetting 조회를 2회로 고정

범위: 개선 수치는 User · UserSetting 조회 기준입니다. 차단 · 기존 관계 조회는 페어마다 남아 있어 후속 개선 대상으로 관리했습니다.

BONE:TOBE

골격 인식 기반 AI 의류 추천 서비스



개요

GitHub ↗ Demo ↗

사용자의 체형(골격)을 AI로 인식하여 체형에 적합한 의류를 추천하고 가상으로 착용해볼 수 있는 웹 서비스입니다.

개발 기간	2026.01 ~ 2026.02 (약 2개월)
플랫폼	Web
개발 인원	팀 프로젝트 6명 (BE 3 + FE 3)
담당	**피드 도메인 백엔드 · 게시글 상세 조회 최적화 · 좋아요 정합성 처리**

담당 기술 스택

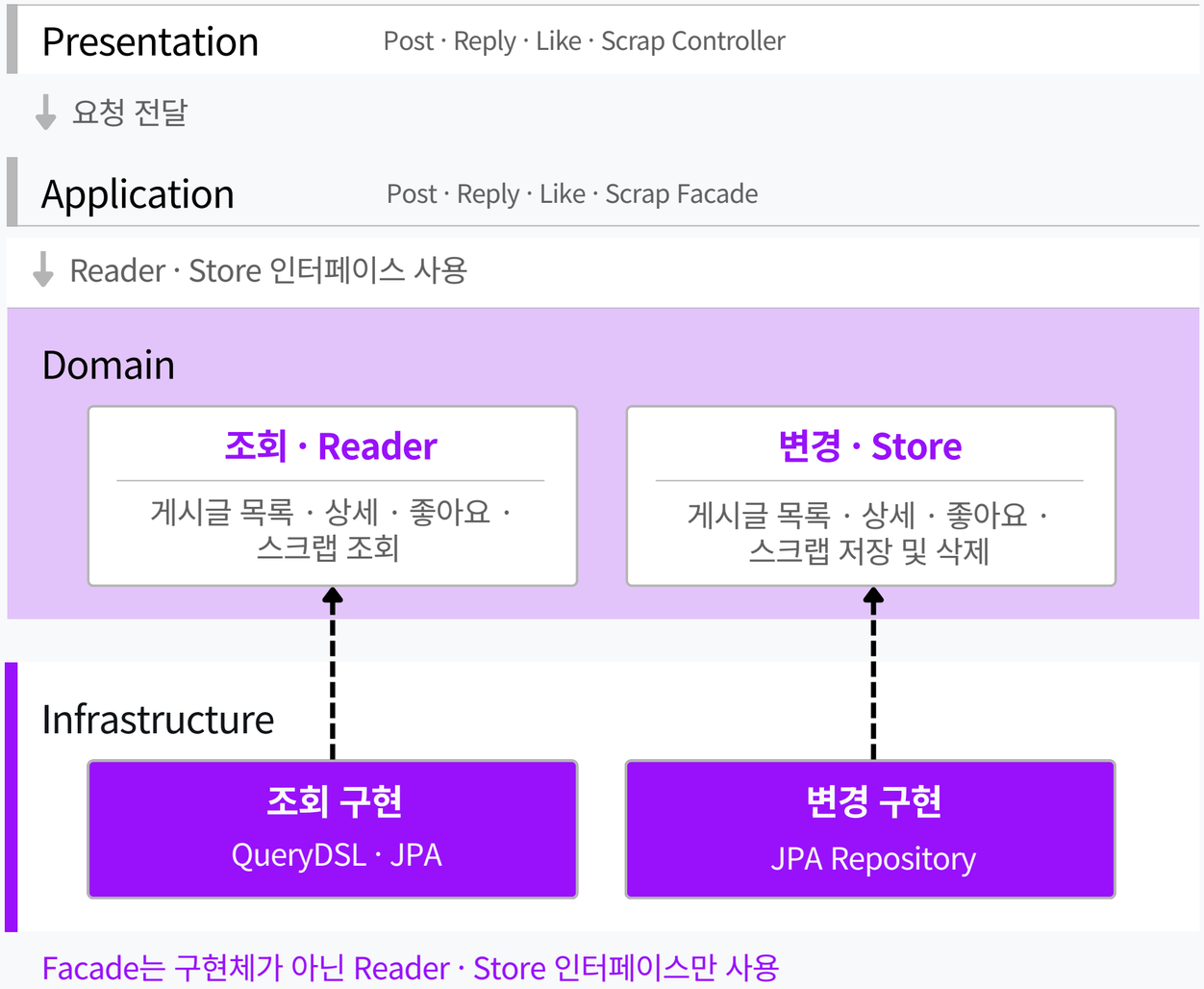
Spring Boot · JPA · QueryDSL · PostgreSQL

- 게시글 · 댓글 · 좋아요 · 스크랩 API 구현
- QueryDSL 기반 상세 조회 통합 · 동적 피드 검색
- 중복 좋아요 요청을 DB 원자 연산으로 처리

아키텍처

피드 기능을 조회와 변경으로 나눈 구조

복잡한 조회와 단순한 변경이 서로 영향을 주지 않도록 역할을 분리했습니다.



Facade는 구현체가 아닌 Reader · Store 인터페이스만 사용

역할 분리

- 조회: 작성자 · 이미지 · 스타일 등 화면에 필요한 데이터를 조립
- 변경: 저장과 삭제에 집중
- 조회 구현이 복잡해져도 변경 로직에 영향을 주지 않음

이 구조를 선택한 이유

- 상세 조회는 QueryDSL로 최적화하고, 변경은 JPA 기반으로 단순하게 유지
- Application은 Reader · Store만 참조해 구현 기술에 직접 의존하지 않음

상세 조회 최적화

게시글 상세 화면에 필요한 연관 데이터를
QueryDSL로 통합 조회하고 성능을 비교·검증했습니다.

기능 · 게시글 상세 통합 조회

상세 화면에 필요한 데이터를 한 번에 조립

작성자 · 이미지 · 스타일 · 좋아요 · 스크랩 상태를 따로 읽지 않고, QueryDSL로 필요한 데이터를 통합 조회했습니다.



파일 조회 포함 엔드포인트 DB 경로 p50 **19.399ms → 7.883ms (59.4% 감소)**

검증: PostgreSQL 15 Testcontainers · warm-up 30회 후 200회 반복 / 로컬 컨테이너 DB 구간 기준

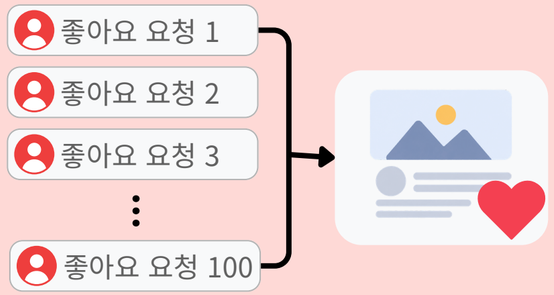
트러블슈팅 · 좋아요 중복 요청

동시에 들어온 좋아요 요청을 한 번만 반영

여러 요청이 동시에 도착해도 DB가 최초 삽입만 성공시키고, 성공한 경우에만 좋아요 수를 증가시켰습니다.

문제

같은 좋아요 요청이 동시에 도착



100개 요청이 모두
‘좋아요 없음’으로 판단할 수 있음

선택

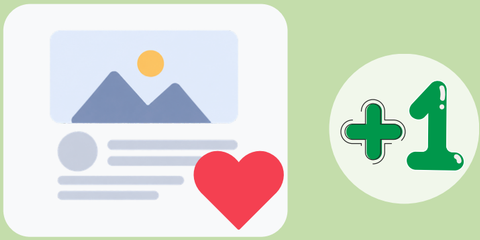
DB가 중복을 한 번만 허용



UNIQUE (post_id, user_id)
INSERT … ON CONFLICT DO NOTHING

결과

성공한 삽입만 카운터 반영



좋아요 행 1건

likeCount 1 · 기타 예외 0



같은 트랜잭션에서 삽입 성공 여부를 확인한 뒤, 성공한 경우에만 likeCount를 증가